Platform for introducing logic programming in bulgarian secondary schools – IDEAS (Intelligent Data Educational Analysis System)

Veneta Tabakova-Komsalova [1,†,*], Magdalena Maglizhanova [1,†,*], Asya Stoyanova-Doycheva[1,†] and Stanimir Stoyanov [1,†]

[1] *University of Plovdiv "Paisii Hilendarski", Plovdiv, Bulgaria*

Abstract

This paper presents an approach to supporting the teaching of logic programming in secondary education. To implement this approach, an educational platform named IDEAS (Intelligent Data Educational Analysis System) has been developed and applied, utilizing C# and Prolog. In addition to facilitating the teaching of logic programming, the platform promotes the development of critical and creative thinking through Project-Based Learning. The role of artificial intelligence in education is also analyzed, highlighting the benefits of its integration and its potential to increase engagement and effectiveness in the learning process. By combining technology and pedagogical methods, the IDEAS platform offers an innovative STEM environment for both students and educators.

Keywords

Education, Prolog, Logic Programming, Artificial Intelligence, Learning Platform, Critical Thinking, Project-Based Learning, STEM education

1. Introduction

Modern education is experiencing a rapidly growing need for innovative methods that foster students' analytical and critical thinking. Logic programming - a paradigm in computer science - offers powerful tools for knowledge modeling and logical inference, making it particularly suitable for project-based and STEM education.

Among logic programming languages, Prolog stands out as an effective tool for solving problems in artificial intelligence, as well as for search, planning, and knowledge representation. Thanks to its resemblance to natural language, its syntax is easily understood by learners of various ages and allows integration into a wide range of secondary school subjects.

Studies such as Cecchi et al. [1] have shown that logic programming is both effective and well-received even among children aged 8–10. Research by Cervoni, Brasseur, and Rohmer [2] demonstrates that learning Prolog in parallel with high school mathematics courses (probability, algebra, calculus, geometry) supports a deeper understanding of mathematical concepts. In Bulgaria, initiatives described by our team in [3] have already introduced Prolog into selected secondary schools. The applications of the language are also expanding into bioinformatics, as shown by the publication of Dal Palù et al. [4], while Marinković [5] summarizes developments of Prolog-based systems for automated geometric reasoning. These examples support the choice of Prolog as a

[1] *Joint Proceedings of the Workshops and Doctoral Consortium of the 41st International Conference on Logic Programming, September 9—13, 2025, Rende, Italy*

* Corresponding author.

† These authors contributed equally.

v.komsalova@uni-plovdiv.bg (V. Tabakova-Komsalova); magdalenamaglijanova@uni-plovdiv.bg (M. Maglizhanova); asstoyanova@uni-plovdiv.bg (A. Stoyanova-Doycheva); stani@uni-plovdiv.bg (S. Stoyanov)

0000–0002–0617–9844 (V. Tabakova-Komsalova); 0000-0002-0129-5002 (A. Stoyanova-Doycheva); 0000–0002–3854–4260 (S. Stoyanov)

fundamental tool for project-oriented and STEM education, enabling Bulgarian schools to join the "Prolog Education" initiative, which is part of the online community "All Things Prolog."
This paper proposes an approach to teaching and learning AI through the introduction of logic programming in Prolog at the secondary school level. It emphasizes two primary forms of instruction —mass and elite education. The paper provides a rationale and justification for the chosen approach. It outlines the core principles of the methodology (project-based and STEM learning; integration of logic programming into school curricula; combining mass and elite approaches; teamwork, etc.). The implementation is realized through the development of a platform and the creation of a network of schools to exchange experience and collaborate on joint project

2. Our Approach to Introducing Logic Programming in Secondary Education

"Non scholae, sed vitae discimus." — Seneca
"Education is not preparation for life; education is life itself." — John Dewey
"Children should be taught not what to think, but how to think." — Margaret Mead

In today's rapidly evolving technological landscape, the messages of Seneca, John Dewey, and Margaret Mead remain strikingly relevant. Education must not merely prepare students for life—it must be life itself, cultivating independence, critical thinking, and creativity. As artificial intelligence (AI) increasingly shapes modern society, the need for new educational approaches becomes urgent.

This section outlines an innovative strategy for integrating logic programming (LP) and AI into secondary education. Our aim is not only to make students users of technology, but active creators —individuals who understand how AI works, how knowledge is represented, and how logic drives intelligent systems. Education must combine technical literacy with abstract thinking, interdisciplinary insight, and ethical responsibility.

Transforming Secondary Education Through AI. AI opens new possibilities for teaching and learning. In this context, a key question arises: will we teach students to use AI, or will we empower them to create it? While AI offers benefits like personalization and automation, its full educational potential lies in teaching students to model, design, and understand intelligent systems. As Seymour Papert noted, computer science concepts can transform how people think and learn [6].

We believe project-based learning (PBL), especially within STEM, is the key. PBL engages students with real-world problems, fosters collaboration, and promotes systems thinking. Through hands-on AI projects, students gain both technical and social skills, as well as a sense of relevance in their learning.

Logic Programming as an Educational Bridge. LP offers a distinct approach rooted in formal logic and declarative problem modeling. Rather than giving step-by-step instructions, students express knowledge as logical facts and rules, letting the system infer solutions. This approach builds structured reasoning, abstraction, and algorithmic thinking.

Prolog, as a core LP language, teaches students to define problems clearly, break them into components, and design efficient logic-based solutions. It forms a natural bridge between mathematics, logic, and AI.

Mass and Elite Education. The question of how to teach LP reflects a broader debate: should we nurture exceptional talent or ensure basic competence for all? The elite model focuses on advanced instruction for high-achieving students, encouraging participation in competitions and early specialization. The mass model emphasizes broad accessibility, real-world applications, and cross-disciplinary use.

Both have merits and limitations. Elite approaches can produce top-level experts but risk excluding latent talent. Mass approaches democratized access but may lack depth. We advocate a combined strategy: foundational LP education for all, with pathways for advanced learners to deepen their skills. This dual-level model supports differentiated learning while avoiding educational inequality.

Integrating LP into STEM. LP can be embedded into computer science courses, developed into specialized modules, or offered through extracurricular activities. Regardless of format, practical experience is essential. Students should see how LP solves real problems, supported by appropriate content and teacher training.

Based on the discussion so far, we can outline the key characteristics of our approach as follows:

1. **Mass and Elite Access** - All students are taught to use AI in their learning, while the most motivated are trained to create AI systems through more complex programs and abstract structures—namely, to develop elementary Prolog programs. The basic level for all students focuses on digital literacy, understanding AI concepts, and applying AI systems. The advanced level targets motivated students, who are encouraged to build their own AI models and participate in specialized programs and competitions. This dual-level approach enables differentiated development and the inclusion of a wide range of learners, while avoiding educational segregation.
2. **Integrating Paradigms.** Rather than opposing Logic Programming (LP) to other paradigms, we highlight the benefits of combining logical, imperative, object-oriented, and functional programming. Students familiar with imperative languages like C# can develop hybrid systems that integrate Prolog-based reasoning [3]. Introducing Functional Programming (FP) in grades 11–12 further enhances abstract and algorithmic thinking through concepts like immutability and higher-order functions [6]. While full-featured FP languages such as F# may be too advanced for most students, lightweight functional features in SWI-Prolog and Ciao Prolog offer accessible entry points [1], [5]. This multi-paradigm strategy aligns with STEM and project-based learning goals, enriching students' understanding of diverse computational models and problem-solving approaches [2], [6].
3. **Teamwork** - We encourage collaboration among students as well as among teachers, fostering a cooperative learning environment.
4. **Network of Schools** - We support the formation of a community of schools that share best practices and jointly build Prolog knowledge bases on topics related to Bulgaria's cultural and historical heritage. These schools may include both creators and users of the Prolog-based AI systems.
5. **Adaptability and Sustainability** - The approach is flexible and applicable both in computer science classes and in interdisciplinary projects. The developed knowledge bases can be used across different subjects that relate to the content of those knowledge bases, supporting cross-curricular integration and long-term use.

Our approach aims not merely to prepare students for the technological future, but to enable their active participation in shaping it. Through logic programming, project-based learning, and a balanced model of mass and elite access, we believe we can contribute to building a smarter, fairer, and more creative educational future. As Malcolm X stated: "The future belongs to those who prepare for it today."

The key principles of effective project-based learning include: a focus on authentic problems that are relevant to students and society; encouragement of deep inquiry and critical thinking; offering students choice and autonomy throughout the learning process; providing multiple opportunities for feedback and revision; and the creation of public products that can be shared with a broader audience. In this context, we select student project topics from the field of Bulgarian cultural and historical heritage. Each project follows the stages outlined in Figure 1.

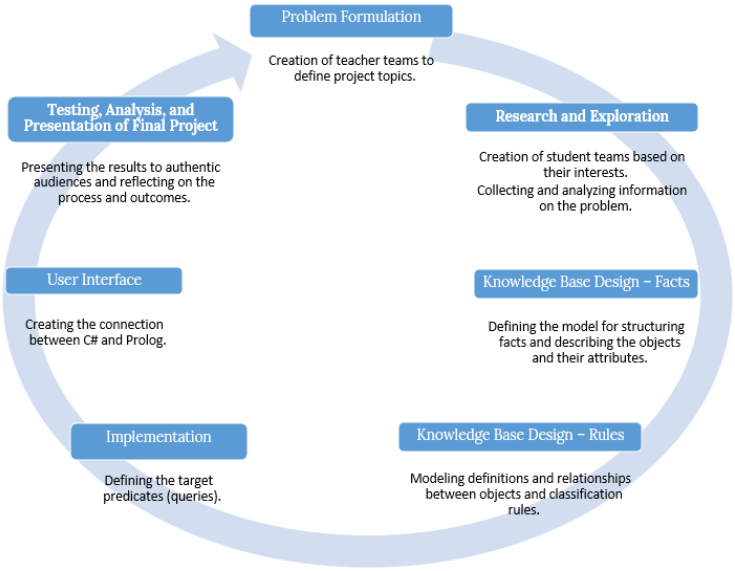

Figure 1. Stages of Project Implementation

Ultimately, project-based learning serves as a powerful tool for preparing students for the future, fostering not only domain-specific knowledge in STEM and AI, but also universal competencies such as critical thinking, problem-solving, creativity, and adaptability—skills that will be essential in the rapidly changing world of the 21st century.

3. The IDEAS Platform

As previously noted, our approach is supported by a platform called IDEAS (Intelligent Data Educational Analysis System). The platform is developed on three levels (Figure 2) – the user level, the management level, and the thematic level.

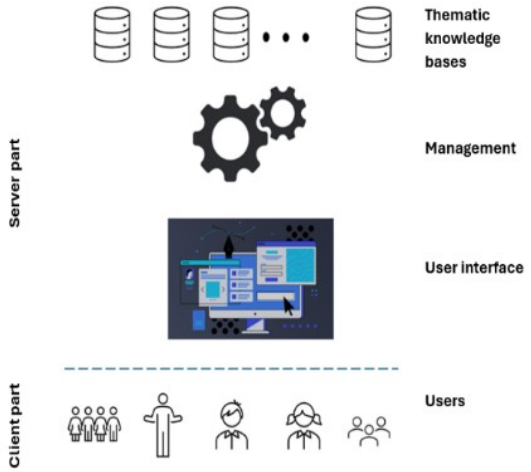

Figure 2. IDEAS Platform Architecture

Thematic Level. At this level, a distributed knowledge base is developed, with each individual sub-base storing specialized knowledge on a selected topic—either from a school subject (biology, chemistry, physics, history, geography, etc.) or an extracurricular STEM theme—represented using the logic programming language Prolog. The assignments for developing these topics are prepared by an interdisciplinary team of teachers. The implementation of the thematic knowledge bases is carried out by student teams who have received prior training in Prolog programming (elite-level training). Each knowledge base is a self-contained component that can be independently activated, updated, or removed. In addition to factual content, these bases may also include predicates to

support system management. The development of individual themes occurs in parallel across the participating schools in the network. Each school conducts local testing of its knowledge bases. Once completed, the final versions are integrated into the distributed knowledge base deployed on the IDEAS server infrastructure. The thematic knowledge is then used for mass-level instruction in the respective subjects. Examples of currently developed thematic knowledge bases include: "Mineral Springs in Bulgaria", "Historical Sites", and "Medicinal Plants."

User Level. This level is served by a specialized user interface. Users of the platform can take on different roles, such as: Learner (accessing resources related to a specific topic or school subject), Content creator (developing a thematic knowledge base), or Administrator. The user interface receives queries from users, passes them to the management layer for processing, and returns the results in natural language text form (Figure 3). The user interface is implemented in C#.

Management Level. This is a virtual network management module that performs several key functions: it analyzes user queries, activates or deactivates thematic knowledge bases, initiates reasoning engines when logical processing is required, and manages resource allocation among network participants. This management layer acts as a mediator between the user interface and the distributed knowledge base. This mediation is carried out in two steps:

- The user selects the topic (subject) of interest to him/her – then the management activates (consults) the corresponding thematic knowledge base.
- The user asks a question – the management activates an appropriate target predicate from the active knowledge base. In our case, we use a relatively simple solution (also taking into account that students can participate in the implementation of the management), which includes observing the appropriate organization and discipline. The creators of thematic bases specify (as comments in Prolog programs) the target predicates and their possible calls. When integrating a thematic base, a correspondence is established between these predicates and the options offered to users through the user interface. These correspondences are used by the management to activate the appropriate targets, depending on the user's request.

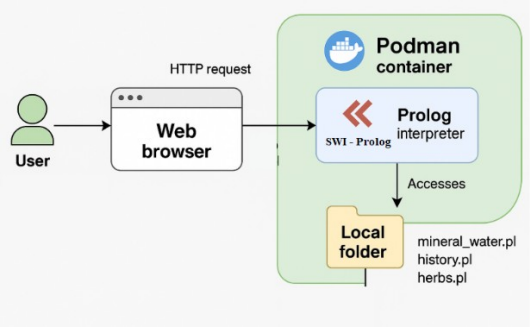

Figure 3. Interaction Between Platform Components

Interaction Between Layers. The platform combines the strengths of logic programming with those of imperative and object-oriented programming, along with the convenience of modern web technologies. This requires coordinated interaction between the different layers of the platform, supported by appropriate configuration of the platform's runtime components. The platform provides a web-based interface (Figure 3) built with C#, giving users direct and convenient access to the distributed knowledge base. Upon launch, users access the system via a web-based chat application, which serves as the primary interface. Through this chat, users can easily select from a predefined set of commands presented as convenient buttons. At the core of the platform are several key components that automate the startup and management of a virtual environment based on Podman, a containerization tool. This virtual environment is pre-configured and includes an installed Prolog interpreter to handle logic-based queries. This integration enables the use of Prolog without requiring complex setup or advanced technical knowledge. The entire process is transparent to the

user: execution and communication with the logical environment are fully managed by the C# code, and the results are displayed directly in the chat interface as responses in natural language.

While the current implementation of the IDEAS platform uses C# for the user interface and web server layer—primarily due to its robust object-oriented features, seamless web integration, and strong development tools—we acknowledge that alternative architectures based entirely on logic programming environments such as SWI-Prolog or Ciao Prolog are technically feasible. These systems offer built-in support for web programming, HTTP handling, and interface design, enabling a fully Prolog-centric development model. While such an approach may reduce the conceptual impedance between the knowledge base and interface layers, and provide a compelling pedagogical example of Prolog's flexibility, we opted for a hybrid architecture to ensure greater accessibility, scalability, and developer familiarity—especially given the context of secondary education. Nevertheless, exploring a unified logic-based stack remains an interesting direction for future iterations of the platform.

4. Demonstration Example

Bulgaria possesses an extraordinary wealth of hot mineral springs scattered across the country. This makes it a highly attractive destination for numerous tourists from all over Europe and the world. Bulgaria ranks second in Europe - after Iceland - in terms of natural mineral and spring water resources, with unique composition and excellent drinking qualities. In addition to their balneological and healing properties, Bulgarian waters are notable for their chemical diversity and are preferred for their natural potability. Nearly all types of naturally occurring waters - based on their physico-chemical characteristics - are represented in Bulgaria. Formed over centuries, these waters are specific to the regions they originate from - such as the Rhodopes, the Balkan Mountains, Rila, and Pirin - and have been used for treating a wide range of ailments since ancient times. The history of using mineral waters in Bulgaria dates back to early civilizations. The Thracians, Greeks, and Romans were among the first to recognize their therapeutic benefits. Roman emperors often visited thermal springs on Bulgarian territory, which led to the establishment of numerous Roman baths and spa towns renowned for their balneotherapy. It is important to note that mineral water in Bulgaria is not merely a tourist attraction, but also a national asset and part of the country's cultural heritage.

For this reason, one of the demonstration knowledge bases developed within the IDEAS platform is dedicated to Mineral Waters of Bulgaria. The following example illustrates our approach. It demonstrates how project-based learning within a STEM context can be used to introduce AI in secondary schools through logic programming in Prolog, following the stages outlined earlier (Figure 1).

The first step involves forming teacher teams. These teams prepare the project topic by identifying a real-world problem of significance—in this case, Mineral Springs in Bulgaria. Due to the interdisciplinary nature of the subject, the team includes teachers of chemistry, physics, biology, history, geography, and informatics. Teachers determine the sources of necessary information.

The second step is forming student teams according to their interests. These teams collect and analyze information about the topic with guidance from the teachers in the respective subjects, and, if needed, consult with domain experts. Based on the collected data, they identify the entities to be included in the knowledge base along with their characteristics—such as name, chemical and physical properties, location, healing effects, etc. A predicate structure is defined for describing these entities and their attributes.

The third step is creating the knowledge base by formulating facts according to the defined model (Figure 4).

```
% Define the springs with their properties
spring(1, 'Belchin-Verila', 46, 354, 65.36, 0, 2.5, [6, 13.83, 122.22, 36.01, 6.11, 5.35, 0, 0], [0.1, 0.06, 94.25, 0.9, 3.61, 0, 0.03, 0.0002]).
spring(2, 'Sudievo', 29, 316.15, 42.8, 0, 0.07, [6.35, 20.15, 23.25, 60.01, 73.22, 6.39, 5, 0.05], [0.05, 0.05, 82.73, 0.62, 0.60, 0.12, 0.03, 0.01]).
spring(3, 'Velingrad-Kamenitza', 87.7, 669.53, 106.46, 0, 3, [10.58, 16.31, 234.35, 6, 140.34, 0.8, 5, 0.05], [0.05, 0.2, 139, 6.72, 8.62, 0.12, 0.03, 0.01]).
spring(4, 'Eleshnitza', 56.6, 276.87, 65.45, 0, 0.34, [8.98, 4.25, 41.97, 39.01, 39.66, 8.52, 5, 0.05], [0.05, 0.05, 66.41, 1.19, 1.4, 0.12, 0.03, 0.01]).
spring(5, 'Bedenski bani', 73.3, 1714.31, 129.20, 154, 1.5, [6.96, 43.26, 329.2, 6, 768.83, 0, 5, 0.05], [0.05, 0.77, 338.20, 29.16, 58.52, 9.73, 0.45, 0.04]).
```

Figure 4. Facts About Mineral Springs

Figure 4 presents facts about mineral springs using the predicate spring/9, whose arguments represent the following attributes: spring ID, name, temperature, total mineralization, silicic acid (H_2SiO_3), carbon dioxide, hydrogen sulfide and sulfides, anions (F^-, Cl^-, SO_4^{2-}, SO_3^{2-}, HCO_3^-, $HSiO_3^-$, NO_3^-, NO_2^-), and cations (NH_4^+, Li^+, Na^+, K^+, Ca^{2+}, Mg^{2+}, Fe^{2+}, Mn^{2+}).

The next step involves defining relationships and associations between the previously described entities. At this stage, students begin modeling definitions of relationships between objects and building classifications using rules. In the presented example, the following classifications were developed:

- Classification of Mineral Waters. Their classification is based on several factors, including temperature, mineral content, and therapeutic benefits. Below is a detailed overview of the classification criteria for Bulgarian mineral waters (Figure 5):
    a. By Temperature: Cold waters: below 20°C. Hypothermal waters: 20°C - 37°C. Thermal waters: 37°C - 50°C. Hyperthermal waters: above 50°C.
    b. By Mineral Content: Low-mineralized: less than 1 g/L. Moderately mineralized: 1 – 10 g/L. Highly mineralized: over 10 g/L.
    c. By Chemical Composition: Bicarbonate waters: rich in bicarbonates. Sulphate waters: rich in sulphates. Chloride waters: rich in chlorides. Sulphur waters: rich in hydrogen sulfide. Iron-rich waters: high in iron content. Silica-rich waters: high in silicon content.
- Therapeutic Classification of Mineral Waters: Drinking waters: intended for internal use, beneficial for treating digestive disorders, kidney conditions, and metabolic diseases. Bathing waters: used externally in spa centers to treat musculoskeletal, neurological, and dermatological conditions. Inhalation waters: applied in inhalation therapy for respiratory illnesses.

```
% Define the classification rules
classify_gas_water(Name, 'Carbonated') :- spring(_, Name, _, _, _, CO2, _, _, _), CO2 > 400.
classify_gas_water(Name, 'Sulfuric') :- spring(_, Name, _, _, _, _, HS, _, _), HS > 10.

classify_mineralized_water(Name, 'Hydrocarbonate') :- spring(_, Name, _, _, _, _, _, Anions, _), member(CO3, Anions), CO3 > 1200.
classify_mineralized_water(Name, 'Sulfate') :- spring(_, Name, _, _, _, _, _, Anions, _), member(SO4, Anions), SO4 > 950.
classify_mineralized_water(Name, 'Sodium-chloride') :- spring(_, Name, _, _, _, _, _, Anions, Kations), member(Cl, Anions),
                                                       member(Na, Kations), Cl > 1200, Na > 400.
classify_mineralized_water(Name, 'Magnesium') :- spring(_, Name, _, _, _, _, _, _, Kations), member(Mg, Kations), Mg > 50.
classify_mineralized_water(Name, 'Sodium') :- spring(_, Name, _, _, _, _, _, _, Kations), member(Na, Kations), Na > 20.
classify_mineralized_water(Name, 'Calcium') :- spring(_, Name, _, _, _, _, _, _, Kations), member(Ca, Kations), Ca > 300.

classify_bio_active(Name, 'Silica') :- spring(_, Name, _, _, H2SiO3, _, _, _, _), H2SiO3 > 50.
classify_bio_active(Name, 'Fluoride') :- spring(_, Name, _, _, _, _, _, Anions, _), member(F, Anions), F > 1500.
classify_bio_active(Name, 'Iron-containing') :- spring(_, Name, _, _, _, _, _, _, Kations), member(Fe, Kations), Fe > 25.
```

Figure 5. Classification Rules by: Gas Composition, Mineral Composition, and Bioactive Element Content

The development of the knowledge base continues incrementally, with various rules implemented according to the outlined classifications.

The following figure (Figure 6) illustrates how the classification rules are structured based on temperature and mineralization.

```
% Classification for temperature
classify_temperature(Name, 'Cold') :- spring(_, Name, Temp, _, _, _, _, _, _), Temp < 20.
classify_temperature(Name, 'Hypothermal') :- spring(_, Name, Temp, _, _, _, _, _, _), Temp >= 20, Temp =< 37.
classify_temperature(Name, 'Thermal') :- spring(_, Name, Temp, _, _, _, _, _, _), Temp > 37, Temp =< 50.
classify_temperature(Name, 'Hyperthermal') :- spring(_, Name, Temp, _, _, _, _, _, _), Temp > 50.

% Classification for mineralization
classify_mineralization(Name, 'Very Low Mineralization') :- spring(_, Name, _, Mineralization, _, _, _, _, _), Mineralization < 50.
classify_mineralization(Name, 'Low Mineralization') :- spring(_, Name, _, Mineralization, _, _, _, _, _), Mineralization >= 50, Mineralization =< 500.
classify_mineralization(Name, 'High Mineralization') :- spring(_, Name, _, Mineralization, _, _, _, _, _), Mineralization > 500.
```

Figure 6. Classification Rules by Temperature and Mineralization

Step by step, the knowledge base is gradually expanded, allowing additional facts and rules to be incorporated. For example, it is possible to define rules for determining the medical benefits of specific mineral waters, as well as for identifying the most suitable spring for certain health conditions using a predefined model, such as: medical_issue(Problem, Water_Type). Furthermore, additional information can be associated with each spring, such as its geographical location and historical data on its usage through the centuries.

At the "Implementation" stage, students define target predicates to be made accessible through the platform's user interface. These are intended for use by both teachers and students (non-programmers) across different subject areas. Examples include: a query by name: classify_spring('Hisariya') or a search by temperature class (Figure 7).

```
% Function to query springs by temperature class
springs_by_temperature_class(TempClass) :-
    classify_temperature(Name, TempClass),
    format('~w is a ~w spring.~n', [Name, TempClass]),
    fail.
springs_by_temperature_class(_).                    ?- springs_by_temperature_class('Thermal').
```

Figure 7. Searching for a Spring by Temperature Class

Since the code was developed by secondary school students as part of a project-based learning initiative, some imperative constructs (such as fail and format(_,_)) were used to aid their understanding of control flow and output formatting. Subsequently, teachers guide the students toward more declarative solutions using predicates like setof/3 and bagof/3, especially when combined with C# for handling presentation and interface logic in the platform' s hybrid architecture.

Through this integrative example, students not only learn and apply the fundamentals of logic programming using the Prolog language but also deepen their understanding of various school subjects such as physics, chemistry, biology, geography, and history. The project is implemented as part of project-based learning within the STEM disciplines studied at school.

Students can work in teams, organized according to their interests and abilities. One group may take responsibility for defining the structure of facts and logical rules, while another group, following the developed model, populates the actual knowledge base. In this way, by involving students with deeper interests and capabilities, the strength of the elite approach is demonstrated. Students not only gain knowledge in the field of technology but also see that their work is meaningful and leads to tangible results.

Students with advanced capabilities (e.g., those already familiar with imperative programming languages) continue working on the project by collaborating with their programming teacher to develop a C#-based interface, establish communication with Prolog, query the knowledge base, and activate the corresponding target predicates (stage: User Interface).

After analysis, testing, and final structuring, the platform is activated and made available for widespread use by different types of users—students, teachers, and educational teams working across various disciplines. A key strength of the platform is that it provides reliable and validated knowledge, an essential and distinguishing feature.

All these steps are carried out within the elite training framework. Our approach aims to make the core concepts of logic programming accessible to students by selecting interested learners and training them to develop intelligent systems. This method emphasizes the practical application of logic programming for solving real-world problems and integrates it into the broader context of STEM education.

Using the IDEAS Platform. Through the developed platform with its intelligent, knowledge-based interface—such as the one built around the topic Mineral Springs in Bulgaria—users can easily and quickly obtain personalized information based on their needs. After submitting a specific question or describing a health issue, the system analyzes the query and, using logical rules, provides a well-founded response (e.g., recommending appropriate mineral waters based on their chemical

composition, temperature, mineralization level, and therapeutic properties). In this way, the platform offers access to expert-level knowledge in a simple and accessible manner—without requiring prior training or technical expertise.

The platform is designed for mass educational use, meaning that students can interact with the intelligent system to acquire new knowledge on topics of interest as part of their regular learning experience. The field labelled "Enter Prolog code or query" serves as the main user input for interacting with the knowledge bases powered by SWI-Prolog. Through this field, users can enter standard Prolog goals such as: classify_spring(X) or spring_medical_issues(X).

The interface allows not only the input of goals but also facts or even rules. However, it is important to note that the entered rules are not saved in the knowledge base, as each query is executed within an isolated SWI-Prolog session. The submitted code does not have a lasting effect on the system's state.

Users can input both simple and compound goals, for example:

spring(_, Name, Temp, _, _, CO2, _, _, _), Temp > 50, CO2 > 100.

This query will extract information from the knowledge base only if the predicate spring/9 already exists in the loaded Prolog session. If a predicate is not defined in the Prolog file or is not loaded, the query will not work—it will return false or an error.

The system includes buttons labelled in natural language, each linked to a corresponding Prolog goal defined by the developer (e.g., "Mineral springs in Bulgaria"). When a button is clicked the system automatically sends the associated Prolog query in the background. The user does not need to understand Prolog syntax or the structure of the knowledge base. This creates an accessible interface for people without technical expertise, bridging the gap between natural language and formal logic.

The term "query" in the context of the system refers to a natural language input that is processed via static templates, represented by buttons. Dynamic interpretation through a large language model (LLM) is not used at this stage. However, integration with an LLM is entirely possible in the future. Such integration would make the system even more accessible to students and a broader audience. We now demonstrate an example of such a mass learning session, walking through a typical interaction with the platform. The user interface is intentionally simplified—designed as a chat application with predefined command buttons that users can easily click.

Upon loading the webpage, a welcome message is displayed from the virtual Prolog assistant (Figure 8). Below the message, users are presented with several ready-made topics to choose from—for example: *Mineral Springs in Bulgaria, Historical Sites, Medicinal Herbs.*

Each selection automatically loads the corresponding Prolog file. After each user query, Prolog returns a response in the chat window. Once a specific knowledge base is selected (Figure 9), the user can press a button to activate a corresponding target query (Figure 10).

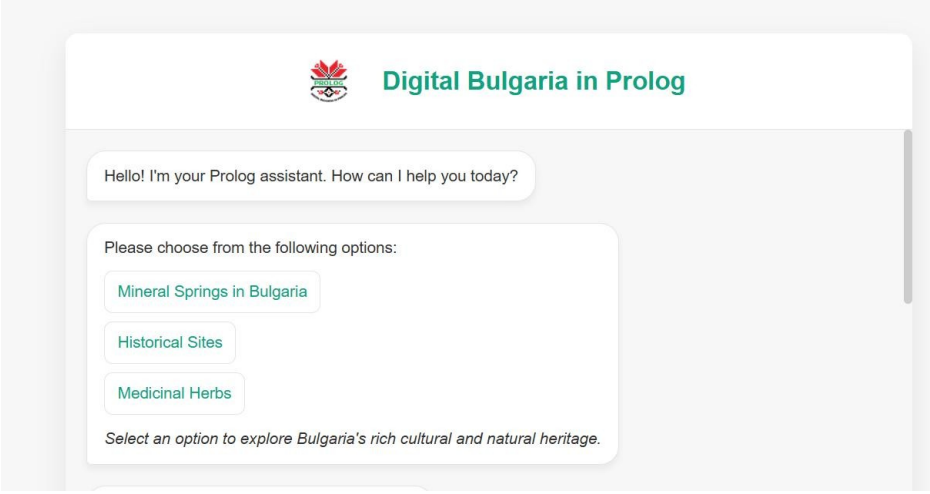

Figure 8. User Interface

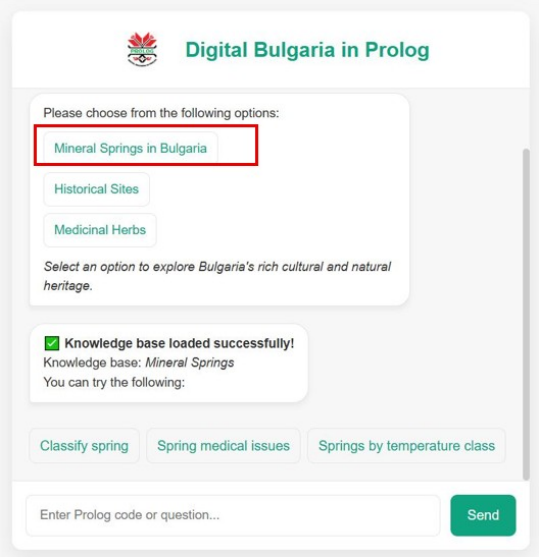

Figure 9. Selection of Thematic Area

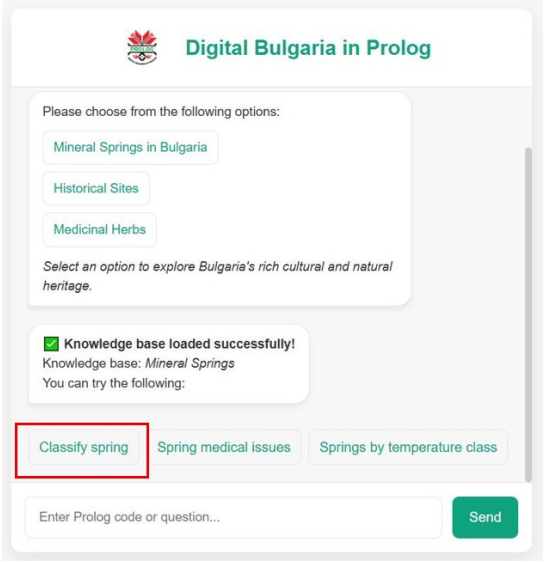

Figure 10. Activation of Target Query

The system then returns a response in the form of natural language text (Figure 11).

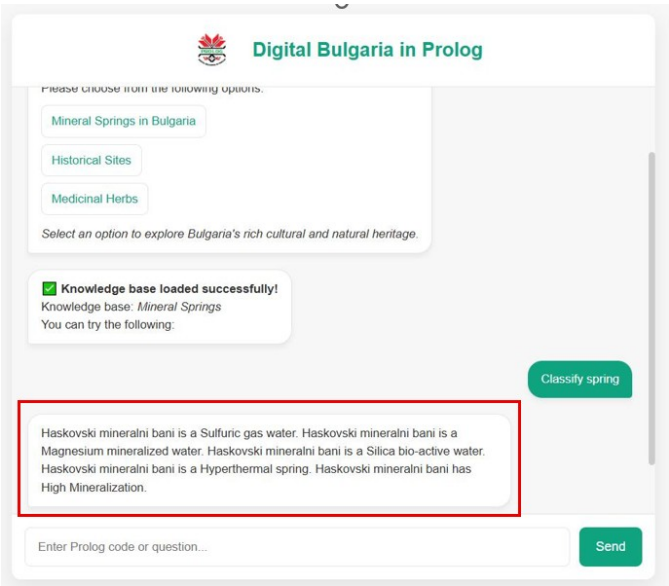

Figure 11. System Response

As previously mentioned, once the knowledge base is structured, a set of so-called target predicates is defined. These predicates determine what types of questions and inferences can be made based on the available information. They form the foundation for query formulation. These queries are created and executed through an intuitive visual interface that does not require any prior programming knowledge. The interface uses buttons to trigger predefined queries linked to the knowledge base. This eliminates the possibility of user error when entering a query. As a result, non-specialist users can easily and reliably use the system for a variety of purposes, enabling its widespread application in educational settings.

5. Conclusions and Future Work

The integration of artificial intelligence and logic programming into STEM education presents both a significant opportunity and a serious challenge. To be successful, it is essential to strike a balance between technological innovation and established pedagogical practices. Ensuring equitable access

to these technologies for all students is critical, along with promoting not only the acquisition of technical skills but also the development of critical thinking, creativity, and ethical responsibility.

The goal is to prepare students not only to use the technologies of the future, but also to create, analyze, and apply them responsibly. In doing so, we ensure that technological progress serves human development—not the other way around.

Creating a school network for logic programming education is an innovative approach to enriching the learning process. An increasing number of schools and teams are joining this network, developing their own knowledge bases. Supported by modern technologies, the network provides an interactive and accessible environment for learning logical principles, which helps students understand more complex topics and encourages analytical thinking. The platform also provides valuable teaching resources to improve instructional practices.

The IDEAS platform may in the future place significant demands on time and computational resources, as student-generated queries may result in high memory usage and long processing times, potentially impacting the response speed of the system.

This integration can significantly boost both motivation and skills among students—key factors in preparing them for careers as future IT professionals. Through the platform and school network, teachers can introduce students to advanced technological concepts and better prepare them for the demands of the digital age [7], [8].

This paper demonstrates the potential of artificial intelligence to transform educational practices, offering innovative methods, tools, and organizational models that can be adapted and applied in various educational contexts. As AI continues to advance, initiatives like this one will play a pivotal role in shaping the future of education, making learning more personalized and effective.

The project will continue in two main directions. The first direction is to expand the network to include new schools, along with additional teacher and student teams. The second direction is extending the platform, including development of the distributed knowledge base with new thematic areas – as the first one is the National Gene Bank. This will allow students to know different plant varieties in Bulgaria. Also, improvement to the management system and user interface, and refinement of the interaction mechanisms between system components.

A fully packaged version of the platform is also planned for deployment on a dedicated server infrastructure hosted at the Faculty of Mathematics and Informatics of the University of Plovdiv.

Acknowledgements

This research was funded by the project КП-06-Н86/9/09.12.2024 „IS-PGR-SADOVO Intelligent system for the management of the Bulgarian plant genetic resources, preserved in the gene bank of IPGR-Sadovo", Fundamental research projects of the National Scientific Fund of the Ministry of Education and Science in Bulgaria.

Declaration on Generative AI

The author(s) have not employed any Generative AI tools.

References

[1] Cecchi, LA, Rodríguez, JP, Dahl, V. (2023). Logic Programming at Elementary School: Why, What and How Should We Teach Logic Programming to Children?. In: Warren, DS, Dahl, V., Eiter, T., Hermenegildo, MV, Kowalski, R., Rossi, F. (eds) Prolog: The Next 50 Years. Lecture Notes in Computer Science(), vol 13900. Springer, Cham. https://doi.org/10.1007/978-3-031-35254-6_11

[2] Laurent Cervoni, Julien Brasseur, Jean Rohmer, Simultaneously Teaching Mathematics and Prolog in School Curricula: A Mutual Benefit, In: Warren, DS, Dahl, V., Eiter, T., Hermenegildo, MV, Kowalski, R., Rossi, F. (eds) Prologue: The Next 50 Years. Lecture Notes

in Computer Science(), vol 13900. Springer, Cham. https://doi.org/10.1007/978-3-031-35254-6_10, Online ISBN978-3-031-35254-6, Online ISBN978-3-031-35254-6

[3] Tabakova-Komsalova, V., Stoyanov, S., Stoyanova-Doycheva, A., Doukovska, L. (2023). Prolog Education in Selected Secondary Schools in Bulgaria. In: Warren, D.S., Dahl, V., Eiter, T., Hermenegildo, M.V., Kowalski, R., Rossi, F. (eds) Prolog: The Next 50 Years. Lecture Notes in Computer Science, 2023, vol 13900. Springer, Cham. https://doi.org/10.1007/978-3-031-35254-6_12

[4] Alessandro Dal Palù, Agostino Dovier, Andrea Formisano, and Enrico Pontelli, Prolog Meets Biology, In: Warren, DS, Dahl, V., Eiter, T., Hermenegildo, MV, Kowalski, R., Rossi, F. (eds) Prologue: The Next 50 Years. Lecture Notes in Computer Science(), vol 13900. Springer, Cham. https://link.springer.com/chapter/10.1007/978-3-031-35254-6_26, Online ISBN978-3-031-35254-6, Online ISBN978-3-031-35254-6

[5] Marinković, Vesna. Prolog in Automated Reasoning in Geometry, In: Warren, DS, Dahl, V., Eiter, T., Hermenegildo, MV, Kowalski, R., Rossi, F. (eds) Prolog: The Next 50 Years. Lecture Notes in Computer Science(), vol 13900. Springer, Cham. https://doi.org/10.1007/978-3-031-35254-6_27, Online ISBN978-3-031-35254-6, Online ISBN978-3-031-35254-6

[6] Papert, Seymour. Mindstorms: children, computers, and powerful ideas, 1980, Library of Congress Cataloging in Publication Data, ISBN: 0-465-04627-4 79-5200.

[7] Meng, N., Dhimolea, T.K., Ali, Z. (2022). AI-Enhanced Education: Teaching and Learning Reimagined. In: Albert, M.V., Lin, L., Spector, M.J., Dunn, L.S. (eds) Bridging Human Intelligence and Artificial Intelligence. Educational Communications and Technology: Issues and Innovations. Springer, Cham. https://doi.org/10.1007/978-3-030-84729-6_7

[8] Ouyang, F., Wu, M., Zheng, L. et al. Integration of artificial intelligence performance prediction and learning analytics to improve student learning in online engineering course. Int J Educ Technol High Educ 20, 4 (2023). https://doi.org/10.1186/s41239-022-00372-4.